QMCPACK Bechmark OLCF-6

This is the OLCF-6 benchmark for QMCPACK.

This repository describes the QMCPACK benchmarks and workflow for OLCF-6. The general OLCF-6 benchmark run rules apply except where explicitly modified in this document and should be reviewed before running this benchmark.

QMCPACK Overview

QMCPACK is an open-source many-body ab initio Diffusion Quantum Monte Carlo code for computing the electronic structure of atoms, molecules, and solids. Ab-initio Quantum Monte-Carlo is one of the leading methods that allow the calculation of many-electron interactions in solids that goes beyond the capabilities of the most widely used density functiona approaches in being able to capture the many body effects that are inaccessible less computationally demanding methods. The QMCPACK code is described and available from https://www.qmcpack.org (https://www.qmcpack.org).

Electron positions are randomly sampled by a large number of Markov chains (or "walkers"). The benchmark uses a diffusion quantum Monte Carlo method that uses a diffusion-and-branch algorithm and a target population of walkers to evolve the walkers in time. During the simulation, the fluctuating population of walkers is load balanced between nodes at every time step of the simulation. In production simulations, measurements/statistics are collected at each time step of the simulation and for all the walkers.

The benchmarks are particularly sensitive to floating point, memory bandwidth and memory latency performance. To obtain high performance, the compiler's ability of optimize and vectorize the application is critical. Strategies to place more of the walker data higher in the memory hierarchy are likely to increase performance.

Code access and compilation

The process of building the benchmark has three basic steps: obtaining the source code, configuring the build system, and compiling the source code.

Obtaining the QMCPACK source Code

The QMCPACK version for this benchmark is the v3.17.1 release from August 25, 2023 and is included with this benchmark package. The source code was originally obtained from QMCPACK GitHub repository at: https://github.com/QMCPACK/qmcpack/releases/tag/v3.17.1 (https://github.com/QMCPACK/qmcpack/releases/tag/v3.17.1)

Configuring the QMCPACK build system

QMCPACK uses Cmake to configure its build. Configuration options are described in the documentation at https://qmcpack.readthedocs.io/en/develop/installation.html#configuration-options (https://qmcpack.readthedocs.io/en/develop/installation.html#configuration-options).

QMCPACK configuration used in Summit benchmarks

QMCPACK version

This benchmark was constructed using the QMCPACK release version 3.17.1 from August 25, 2023.

Software prerequisites

Building QMCPACK has the following compiler and library requirement:

- C/C++ compiler. **C++ compilers are required to support the C++ 17 standard.**
- An MPI library.
- BLAS/LAPACK, numerical, and linear algebra libraries. Use platform-optimized libraries where available.
- CMake, build utility (http://www.cmake.org (http://www.cmake.org)).
- Libxml2, XML parser (http://xmlsoft.org (http://xmlsoft.org)).
- HDF5, portable I/O library (http://www.hdfgroup.org/HDF5/ (http://www.hdfgroup.org/HDF5/)). Good performance at large scale requires parallel version >= 1.10.
- BOOST, peer-reviewed portable C++ source libraries (http://www.boost.org (http://www.boost.org)). Minimum version is 1.61.0.
- FFTW, FFT library (http://www.fftw.org/ (http://www.fftw.org/)).

Building QMCPACK

When using the QMCPACK distribution downloaded from https://github.com/QMCPACK/qmcpack/archive/refs/tags/v3.17.1.tar.gz (https://github.com/QMCPACK/qmcpack/archive/refs/tags/v3.17.1.tar.gz), follow these steps:

```
tar -xzf qmcpack-3.17.1-tar.gz
cd qmcpack-3.17.1/build
cmake -DCMAKE_C_COMPILER=mpicc \
      -DCMAKE_CXX_COMPILER=mpicxx \
      ..
make -j 8
```

Possible configuration options can be found in the QMCPACK manual at https://qmcpack.readthedocs.io/en/develop/installation.html#configuration-options (https://qmcpack.readthedocs.io/en/develop/installation.html#configuration-options).

Running the benchmark

Input files and batch scipts for the various systems are provided in the benchmarks directory. Each problem has its own subdirectory within the benchmarks directory. Benchmark problems with three different base memory requirements are provided (low [32GB], mid [64GB], high [96GB]). These problems can be run with different number of walkers per node or device and the best performance number obtainable should be reported.

The input files for the benchmark runs are the pseudopotentials (`Ni.opt.xml` and `O.xml`), the orbital file (`NiO-fcc-supertwist111-supershift000-S64.h5`) and the main qmcpack input (`NiO_256_low.in.xml`, `NiO_256_mid.in.xml` and `NiO_256_high.in.xml`). To run the benchmark, `$walkers` in the input file has to be replaced by the number of walkers per MPI rank and `$orbpath` with the path to the orbital file.

Within the respective benchmark directory run the job through the respective equivalent of

```
srun -n #mpi_ranks /path/to/qmcpack/qmcpack --enable-timers=fine NiO-example.in.xml >qmc.out
```

(Example script for automatic replacement and job submission on OLCF Summit for multiple walker counts are provided as `submit_runs-nodes_1.py` and `submit_runs-nodes_54.py`)

The provided script `qmc_throughput.py` extracts the troughput number for a benchmark run from the QMCPACK stdout, i.e. the `qmc.out` from above.

The results reported should be **monte carlo steps per second for the proposed system**, **monte carlo steps per second for a single node** and **number of walkers per node** used to achieve these results.

Benchmark	Size	min. device memory
NiO_256_low	Small	32GB
NiO_256_mid	Medium	64GB
NiO_256_high	Large	96GB

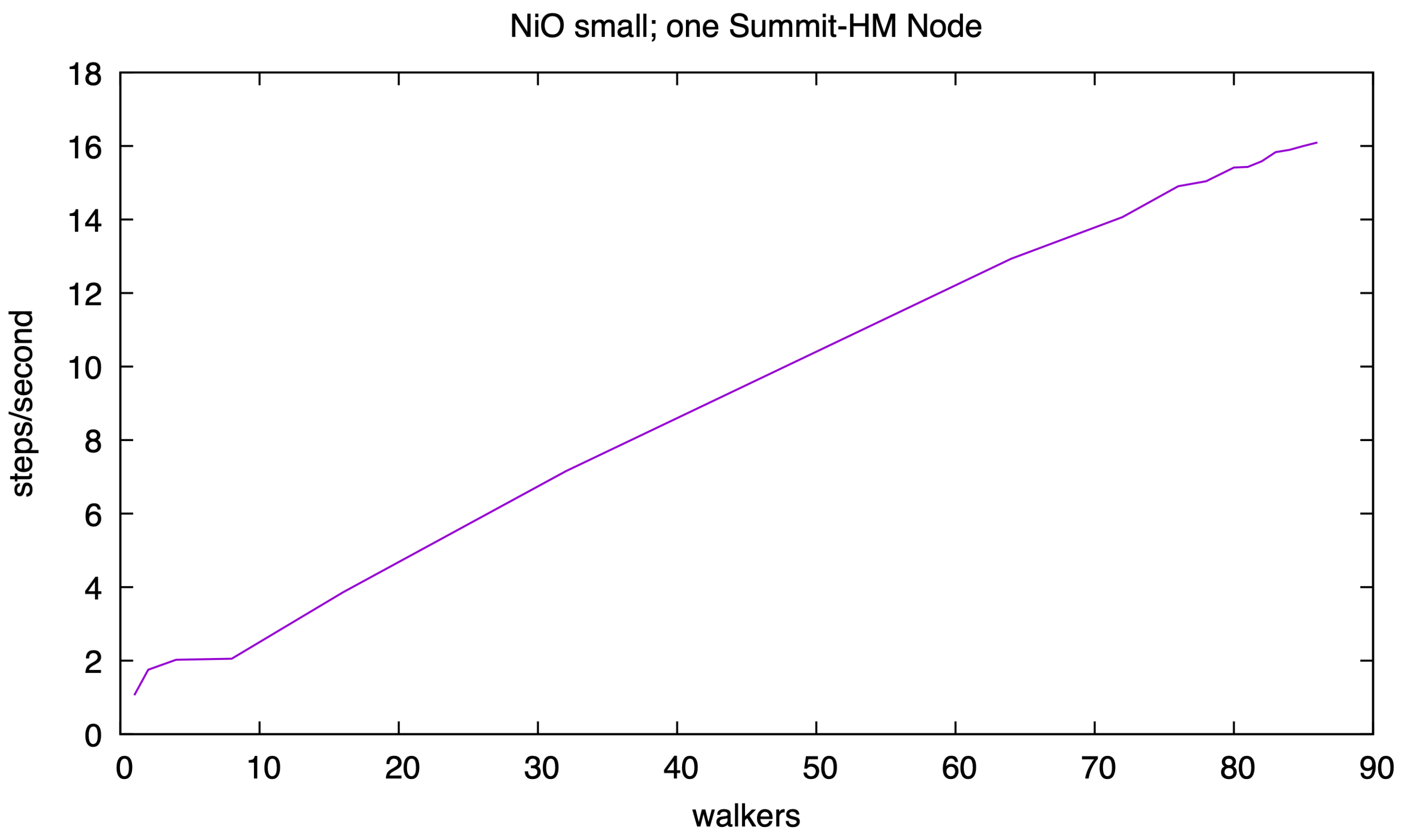
Results

Reference performance: OLCF Summit

As the regular compute nodes only provide 16GB of device memory, the benchmark references on Summit are limited to the use of the Summit High Memory nodes that provide 32GB of device memory. As there are only 54 HM nodes available, here we report the reference performance for 1, 4 and 54 nodes for the NiO_256_low case.

Benchmark.	Nodes	Walkers / rank	Steps/second	Steps/second / node
NiO_256_low	1	86	16.095	16.095
NiO_256_low	4	86	64.643	16.161
NiO_256_low	54	86	871.981	16.148

As ilustration we provide the steps/second on a single Summit-HM node up to the memory limit.


NiO small; one Summit-HM Node
steps/second
walkers
0
2
4
6
8
10
12
14
16
18
0
10
20
30
40
50
60
70
80
90